

1/10

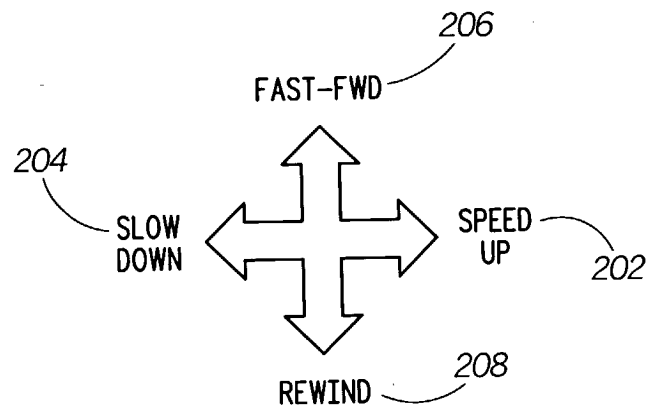
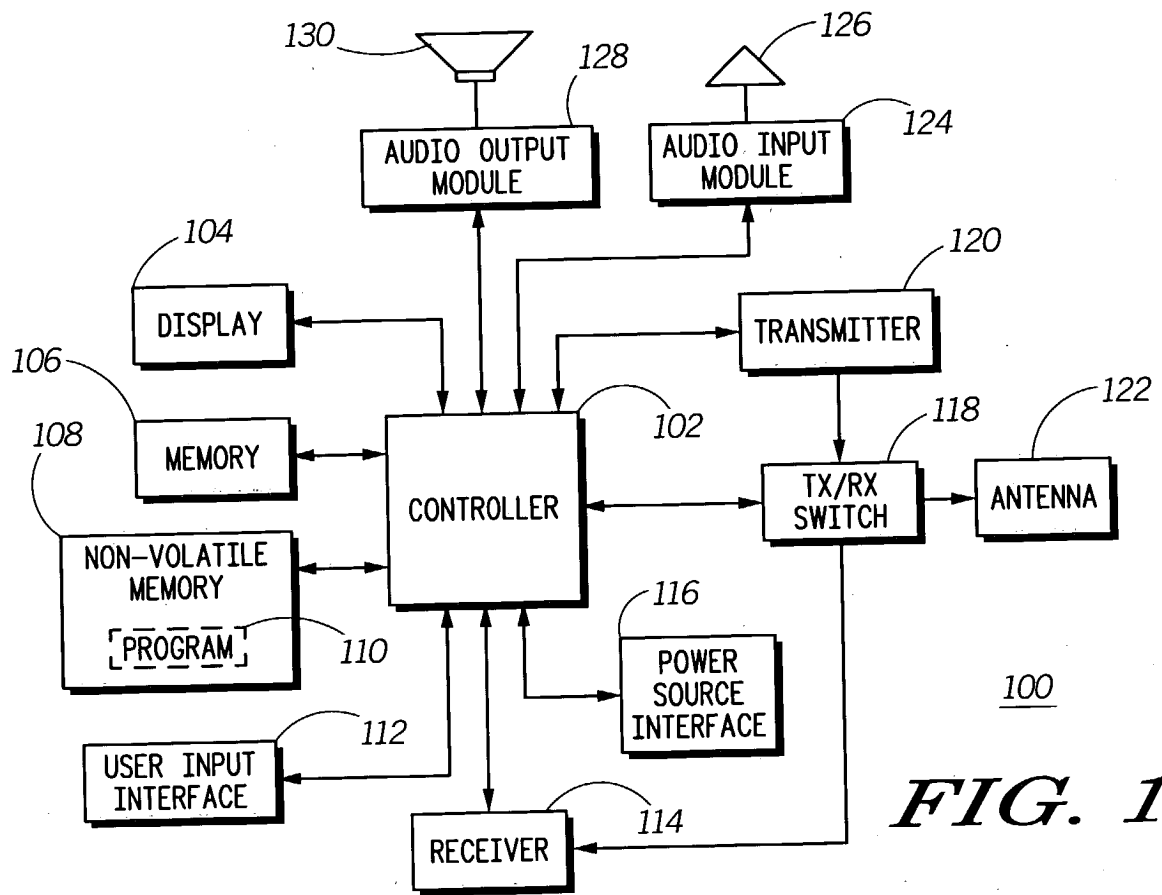


FIG. 2

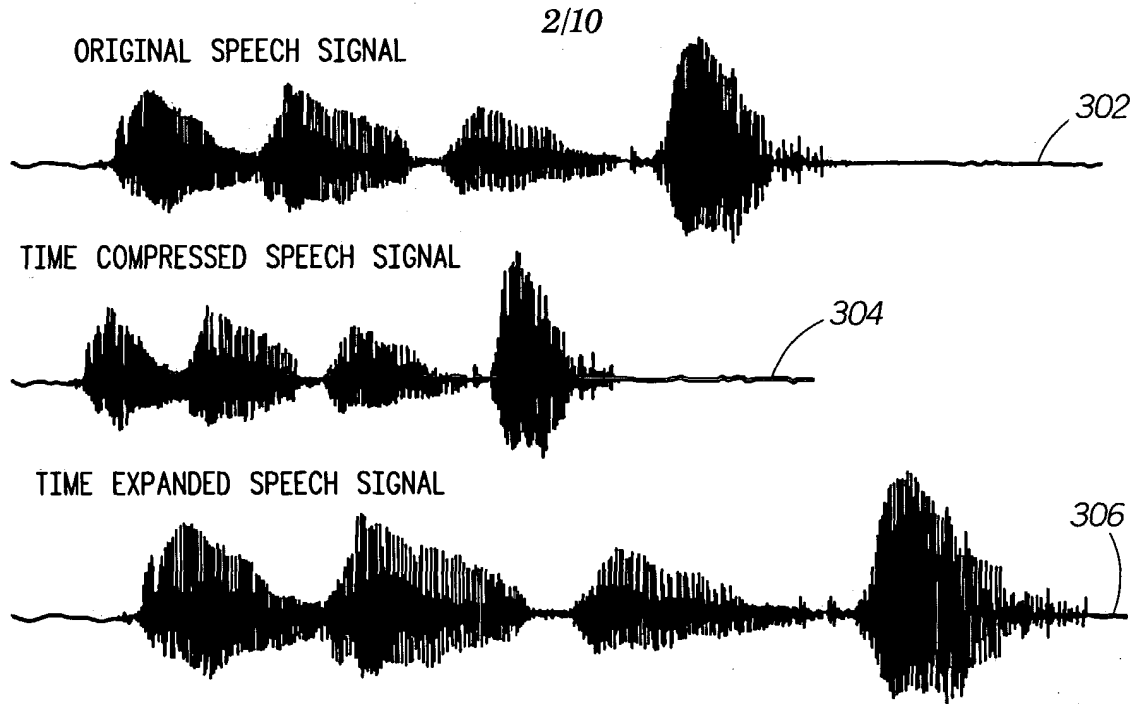


FIG. 3

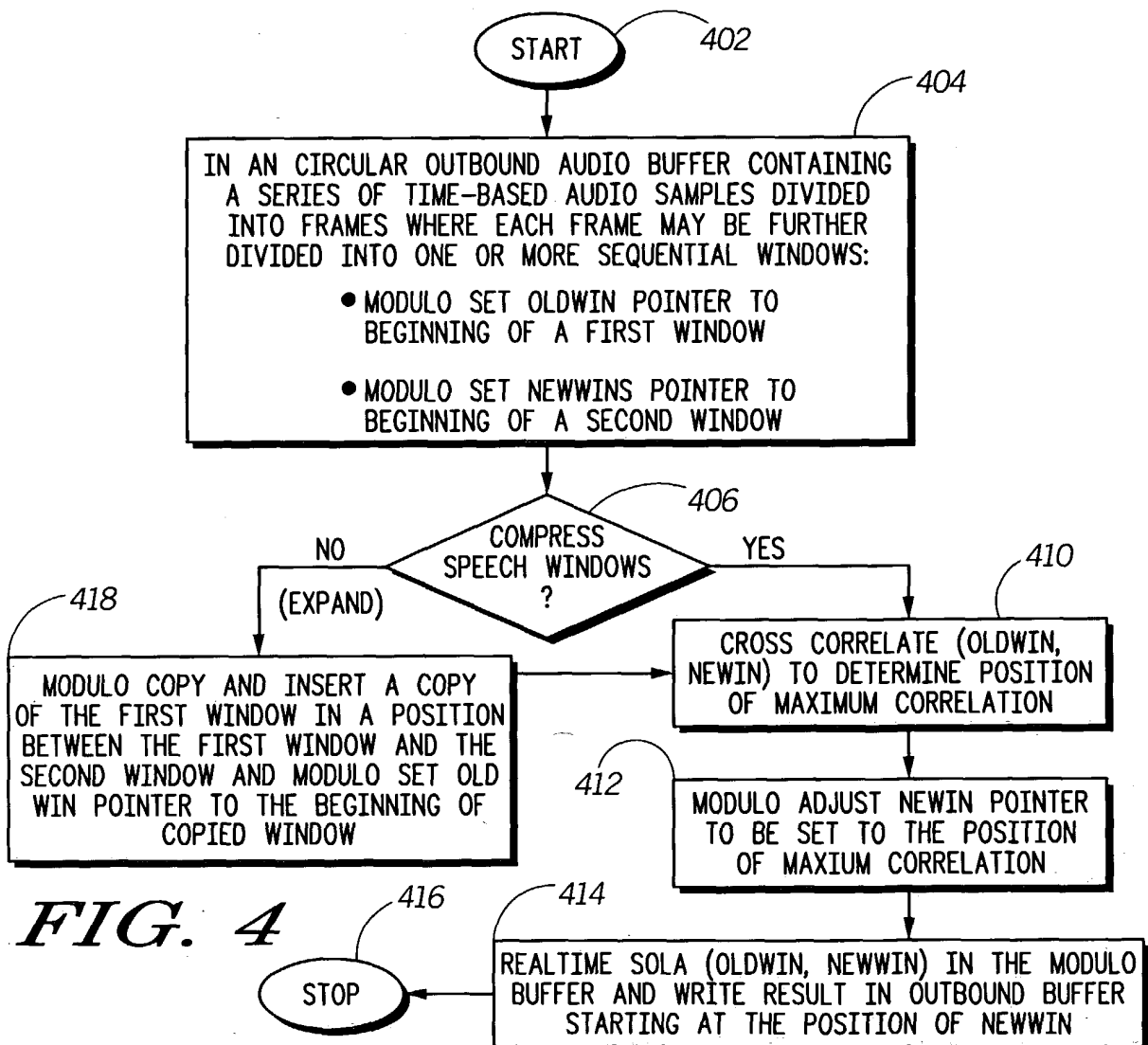


FIG. 4

3/10

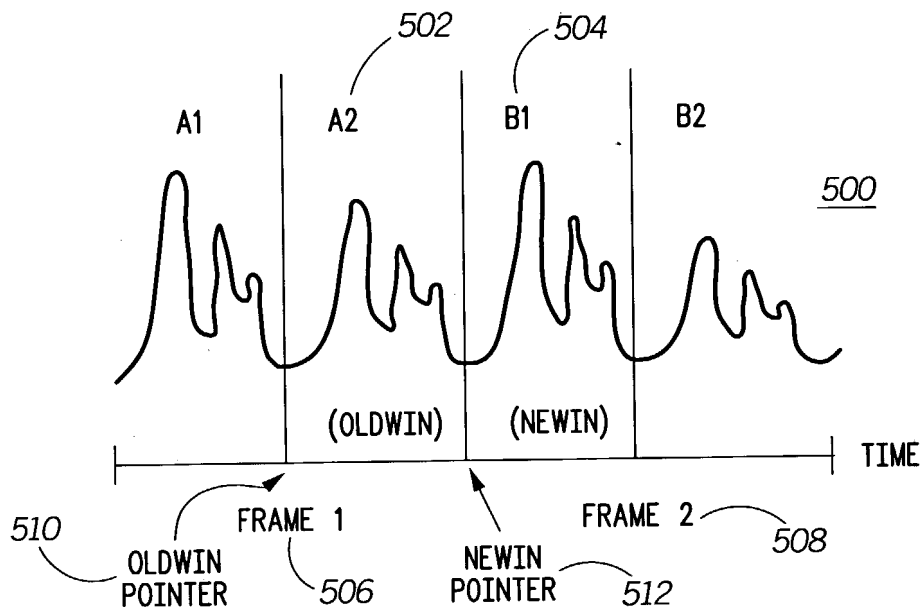


FIG. 5

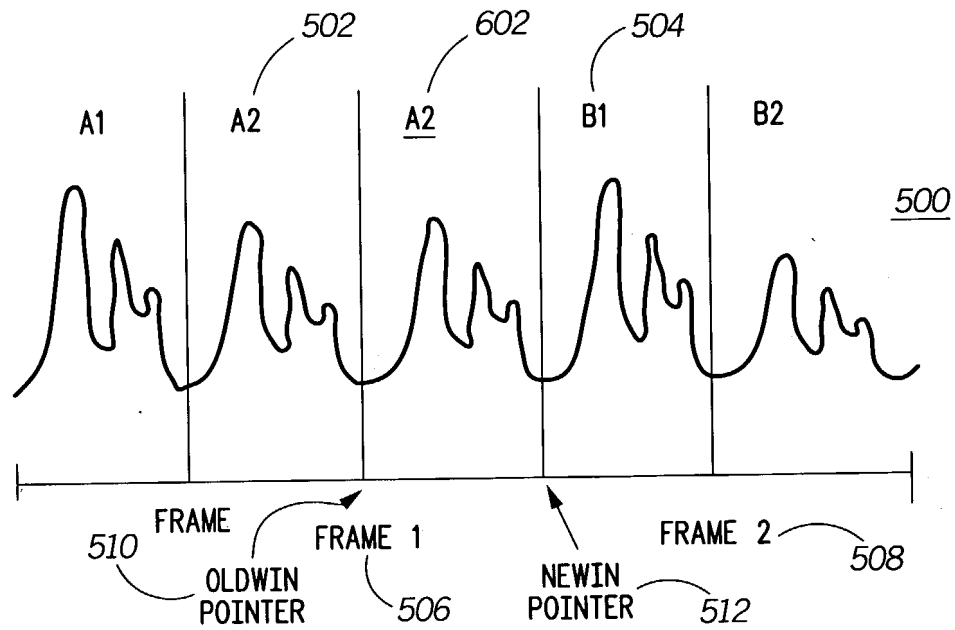


FIG. 6

4/10

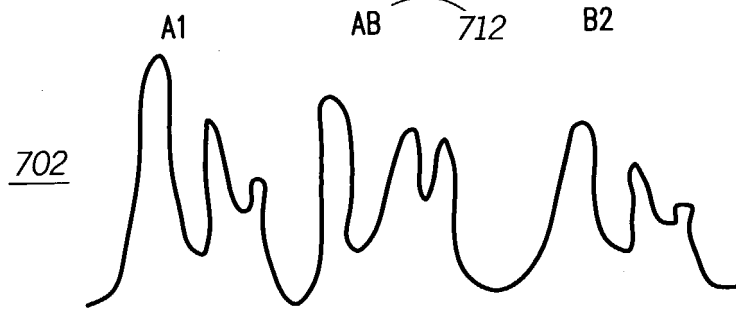
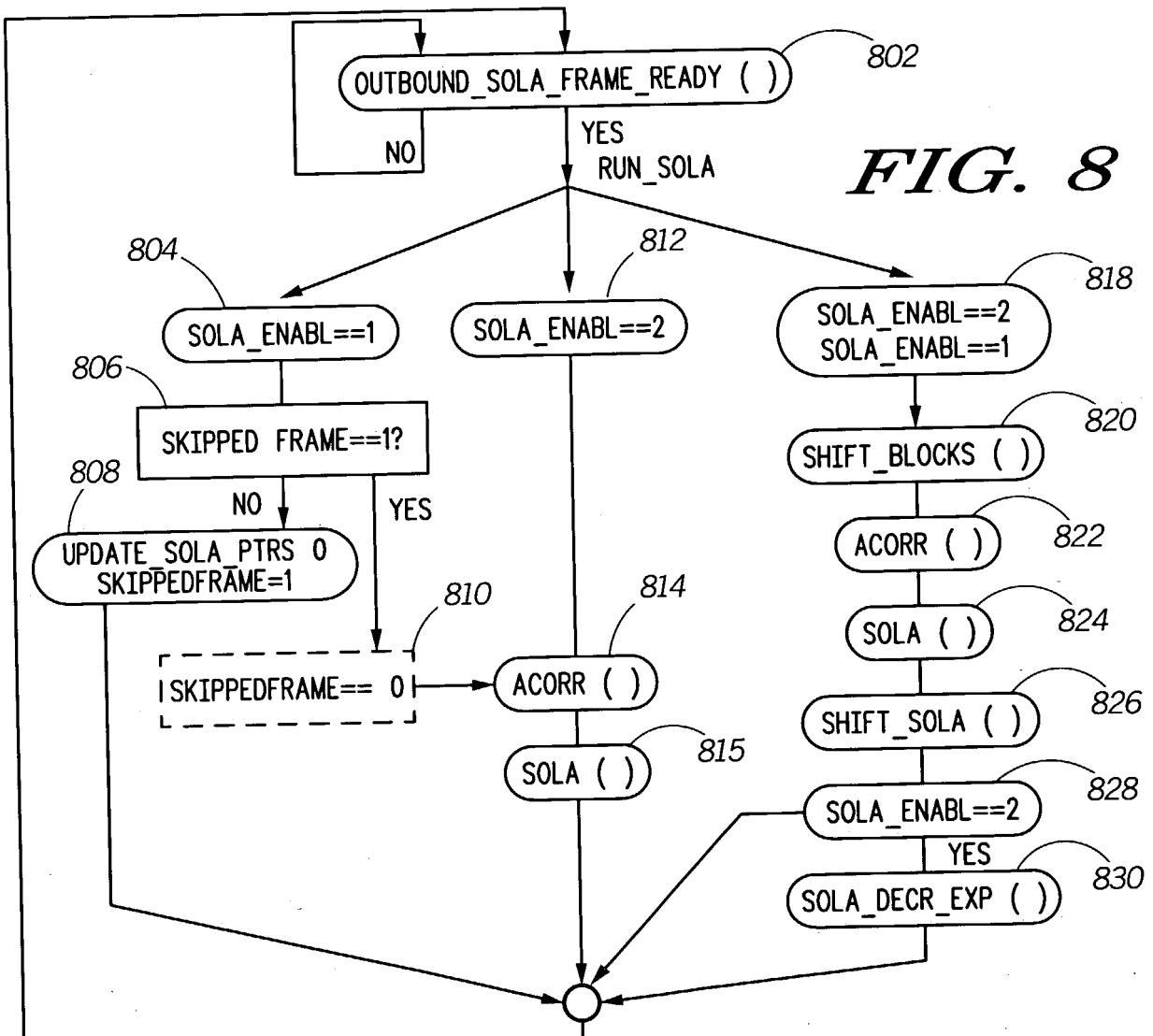
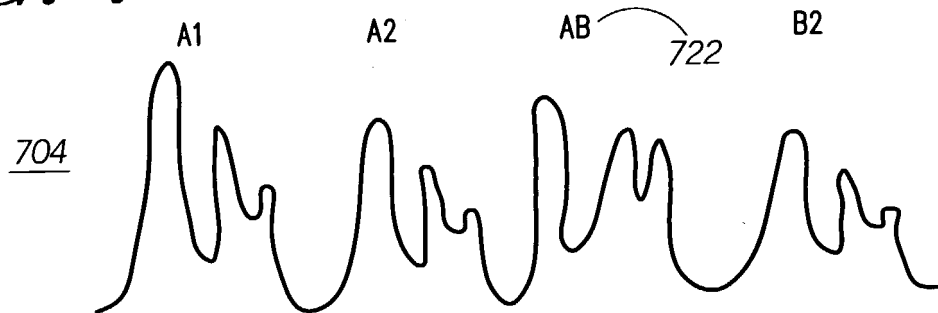


FIG. 7



5/10

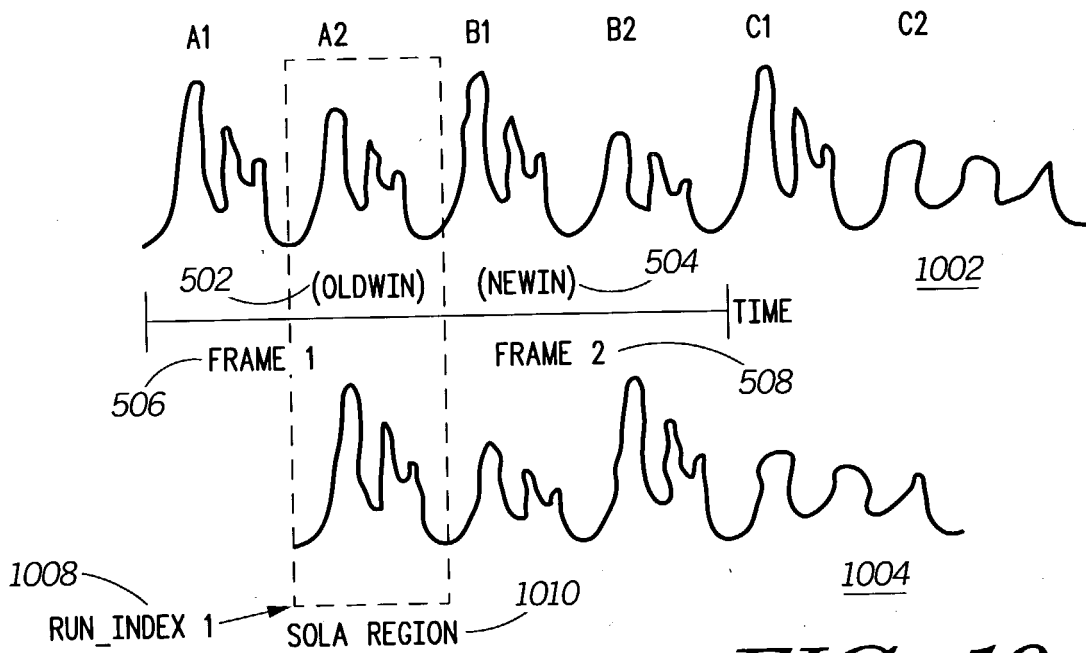
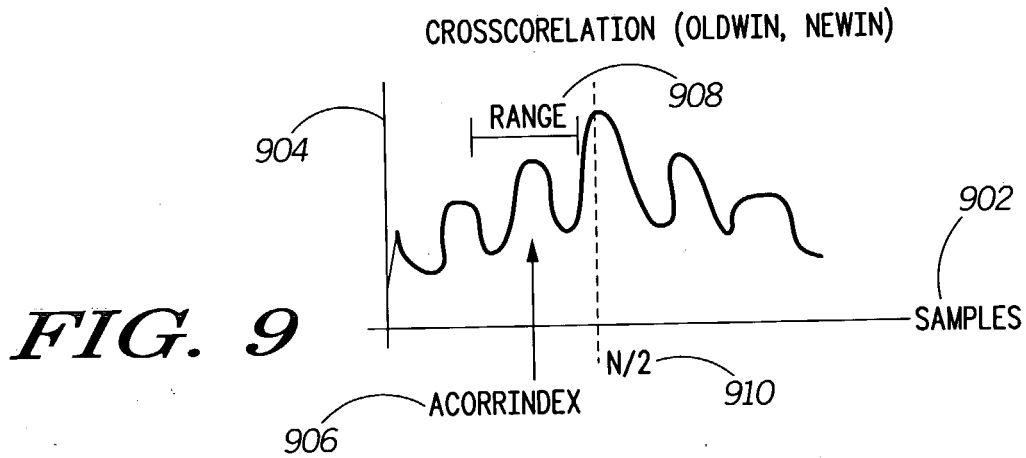


FIG. 10

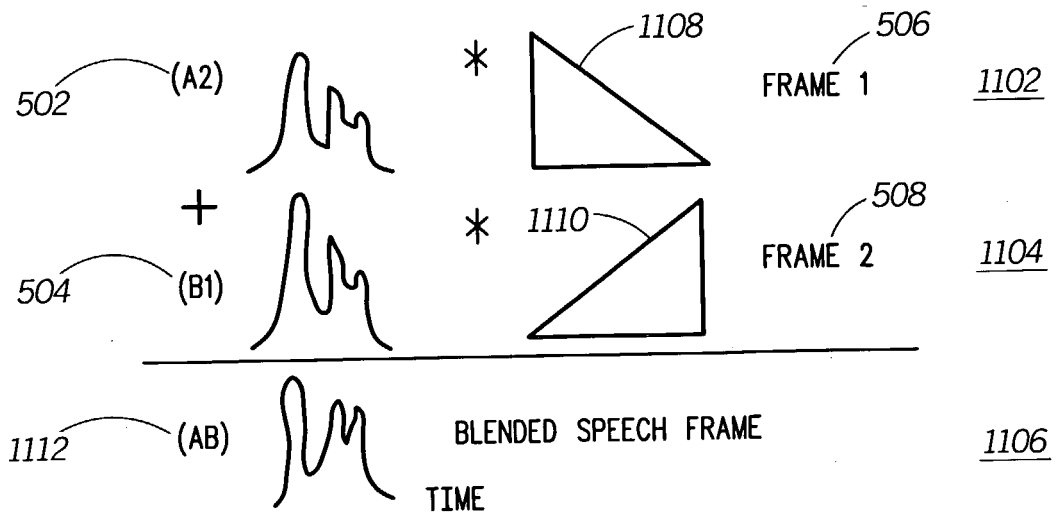


FIG. 11

6/10

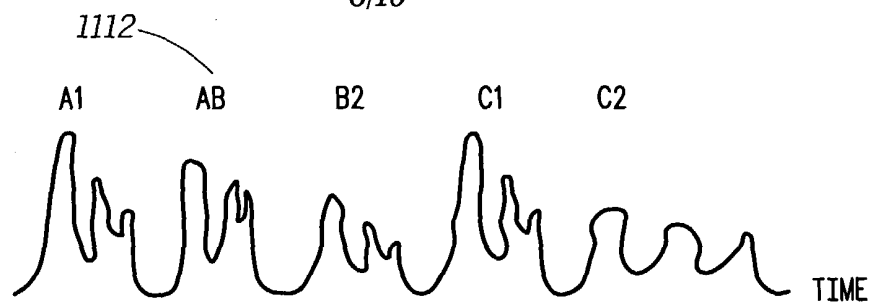


FIG. 12

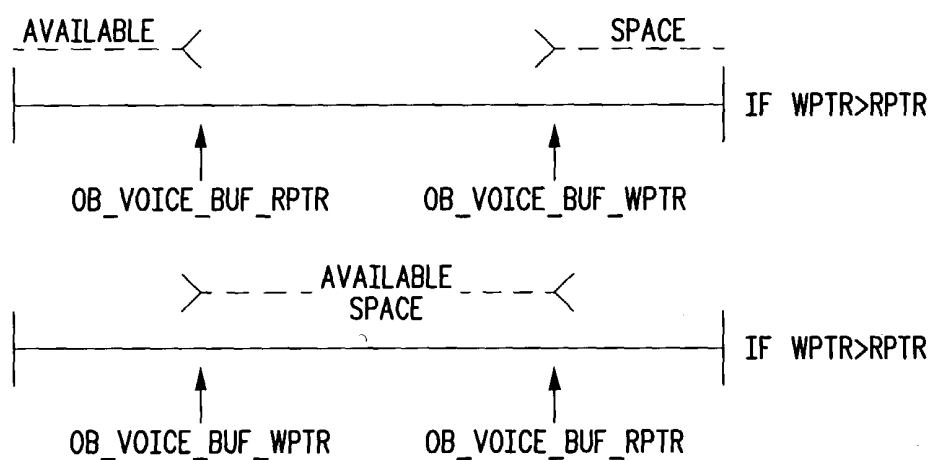
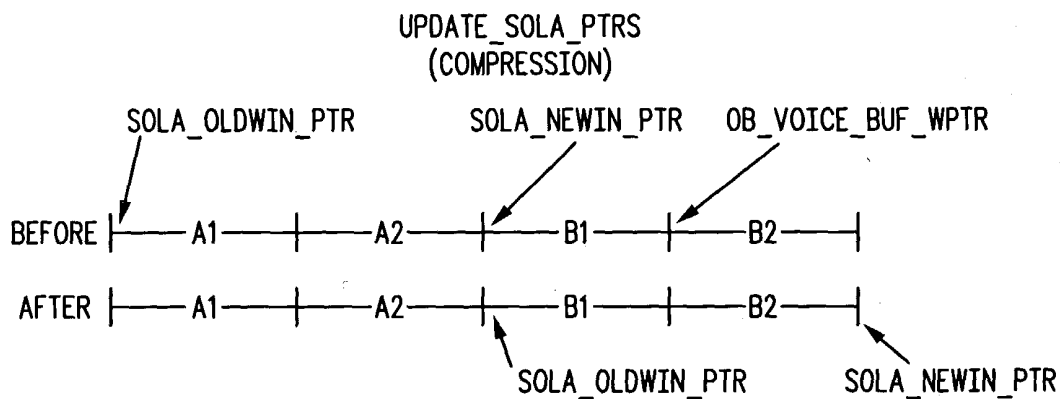


FIG. 13



OUTBOUND AUDIO BUFFER 1024 SAMPLES

1 FRAME IS [A1 A2] OF LENGTH N WHERE A1 IS THE FIRST HALFFRAME

FIG. 14

7/10

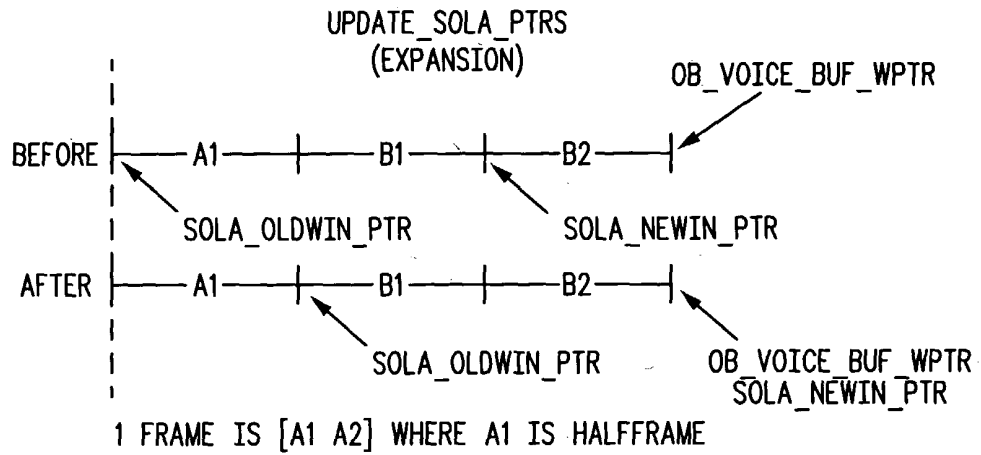


FIG. 15

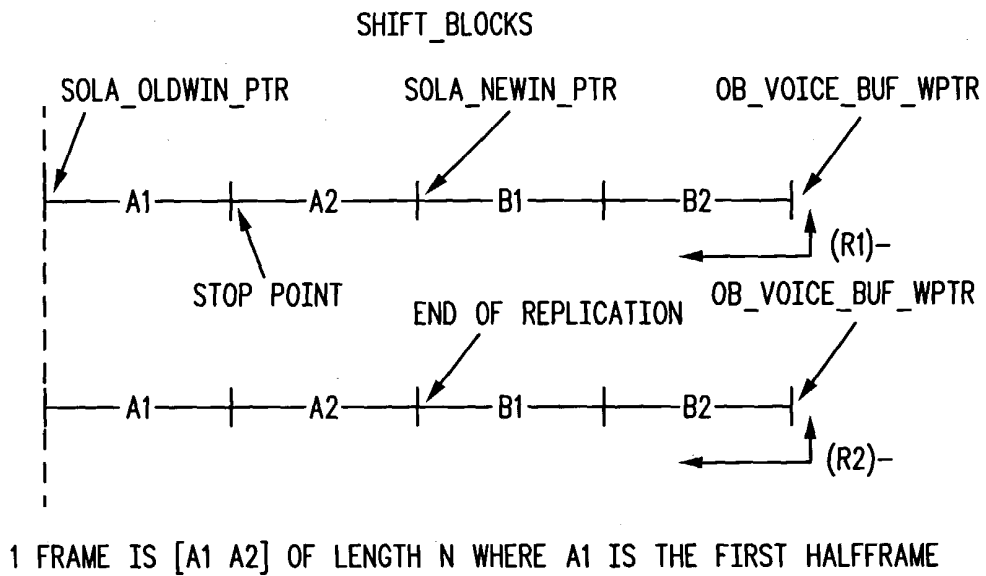


FIG. 16

8/10

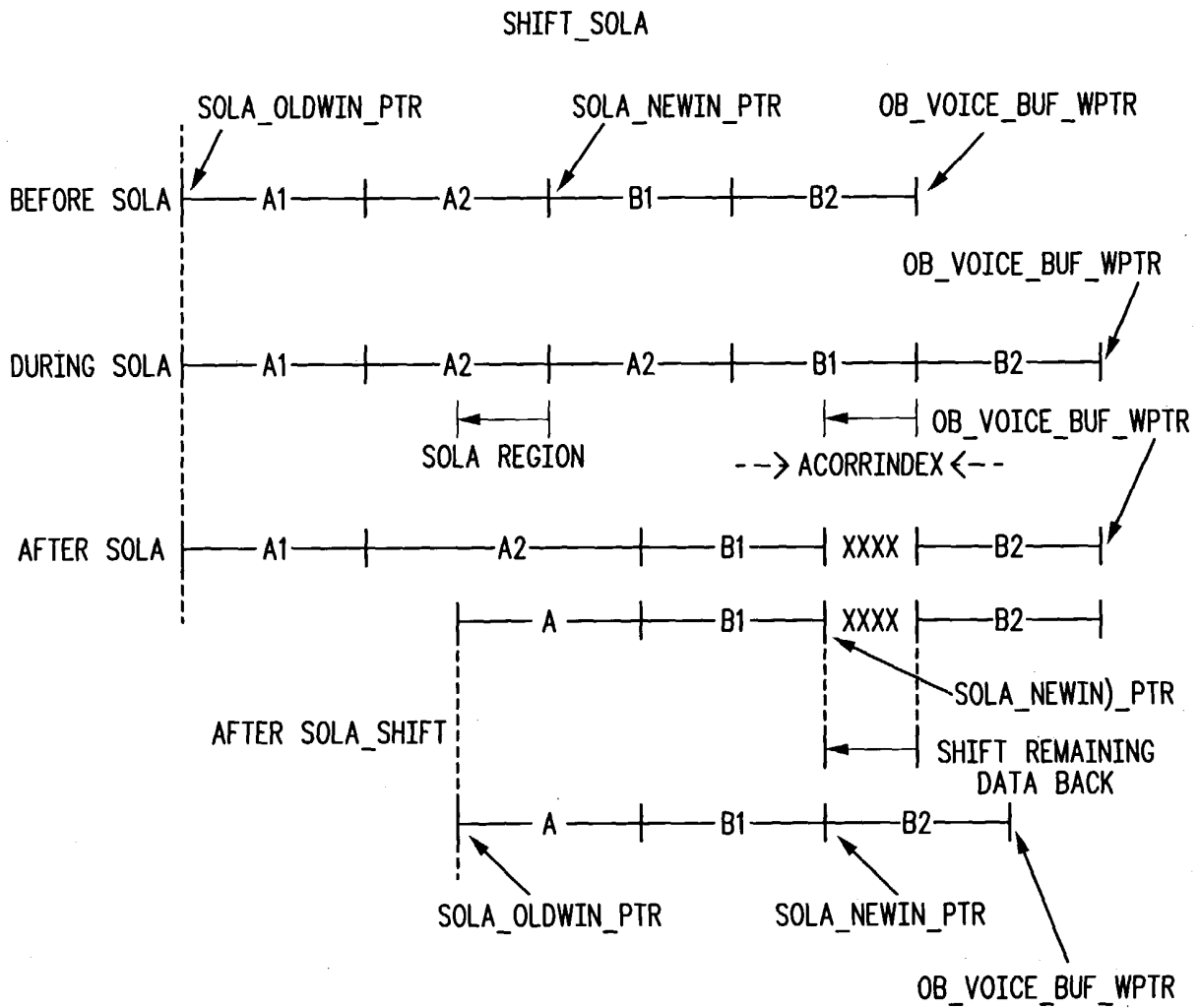


FIG. 17

9/10

```
%-----
% TEMPLATE FOR ASSEMBLY CODING
%
% Implement temporal compression sola algorithm straight up
% It simply processes frame by on a real time basis
%
% CALLS SOLA 0
%-----

clear
path(path,'d;\sola\asm_sola3');
x=hexreadib('scott1.io',0);

global y oldwin;
global winlen winstep cc_range;
global runindex1 runindex2;
global cc_valsI cc_vals2

Nframes=13;
compress=0;

    % convert time of window length to number of samples winlen = 180;
    if(compress)
        winstep=winlen;
        nwins=Nframes;
        cc_range=winlen/2; %compression
    else
        winstep=winlen/2;
        nwins=2*Nframes;
        cc_range=winstep/2; %expansion (==winlen/4)
    end
    % compute total munber of windows in input vector
    % cc_range is used to set cross-correlation detection range

oldwin=x(1:winlen);
y=oldwin;
runindex1=1;
runindex2=winlen;

for i=2:nwins
    begin=(i-1)*winstep+1;
    frame=x(begin; begin+wilen-1);
    y=sola(frame);
end
```

FIG. 18

10/10

```
%-----
%
% This is an implementation of the synchronized overlap and ad method (SOLA)
% for time scale compression
%
% SPECIFICALLY FOR USE AS A REAL TIME APP WHICH ACCEPTS 1 FRAME PER
% X-should be one frame of speech only
% Y-storage be one frame of speech only
% cf-compression factor
%
%-----
```

```
function [y]=sola(newwin)

global y;
global winlen winstep cc_range;
global runindex1 runindex2;
global cc_vals1 cc_vals2

    % compute the cross-correlation
    the_corr=xcorr(oldwin, newwin);

    [maxcc,indx]= max(the_corr(:cc_range));

    runindex2=winlen+runindex1 -1; %always end of old frame

    runindex1=runindex2-indx+1;
    grad=[indx]/(indx+1);

    % sola region
    y(runindex1:runindex2)=y(runindex1:runindex2).*flipud(grad)...
    +newwin(:indx).*grad;

    % append remainder of new frame
    :runindex1+winlen-1=newwin((indx+1):winlen);
    oldwin=y(runindex1:runindex]+winlen-1);
```

FIG. 19

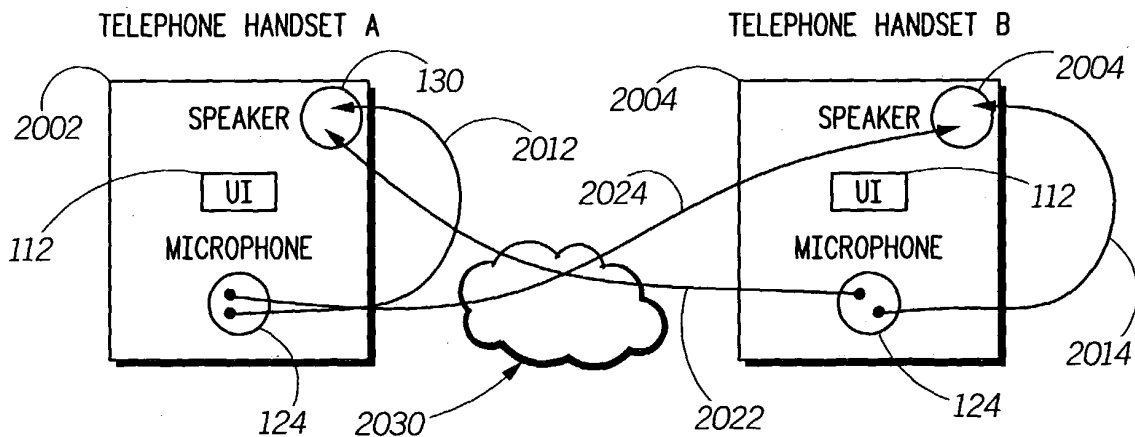


FIG. 20